МІНІСТЕРСТВО ОСВІТИ І НАУКИ,
МОЛОДІ ТА СПОРТУ УКРАЇНИ

Національний технічний університет
«Харківський політехнічний інститут»

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторної роботи
«Процедури у програмах мовою Delphi»

з курсу «Програмування»
для студентів напряму 6.040302 – Інформатика
(спеціалізація «Соціальна інформатика»)

Затверджено редакційно-видавничою
радою університету,
протокол № 2 від 01.12.10.

Харків НТУ «ХПІ» 2011

Методичні вказівки до лабораторної роботи «Процедури у програмах мовою Delphi» з курсу «Програмування» для студентів напряму 6.040302 – Інформатика (спеціалізація «Соціальна інформатика») / Уклад. М. І. Безменов. – Х. : НТУ «ХПІ», 2011. – 16 с.

Укладач М. І. Безменов

Рецензент Л. М. Любчик

Кафедра системного аналізу і управління

ВСТУП

Одним із засобів, використання яких полегшує розробку більш менш складних программ, є підпрограми, зокрема, – процедури.

Мета роботи – освоєння методики визначення та практичного застосування процедур у програмах, написаних мовою Delphi.

1. ТЕОРЕТИЧНІ ОСНОВИ

1.1. Найпростіші процедури

Якщо результат роботи підпрограми не може бути записаний у одну змінну, то таку підпрограму звичайно оформлюють як процедуру. Так само оформлюють підпрограми, які по закінченні своєї роботи взагалі не повинні повертати жодного значення (наприклад, підпрограма для виводу таблиці).

Опис процедури є подібним до опису функції. По суті, існують тільки такі відмінності:

- у заголовку замість службового слова **`function`** вживається слово **`procedure`**;
- у заголовку відсутній тип імені підпрограми;
- у тілі процедури не виконується присвоювання значення або змінній `Result` (ця змінна в процедурі взагалі відсутня).

За винятком цих моментів і методики виклику, все що стосується методики опису функцій може бути перенесене і на процедури.

Важливим моментом при використанні процедур є застосування так званих **`var`**-параметрів. Якщо в процесі роботи необхідно змінити значення декількох змінних, то відповідні імена можуть бути не згадані в списку формальних параметрів (тобто відповідати глобальним змінним). Однак можна в списку формальних параметрів перед деякими іменами задати службове слово **`var`**, яке свідчить про те, що дані імена визначають змінні, які змінюють свої значення в процесі роботи підпрограми. У списку формальних параметрів процедури можуть зустрічатися в будь-якому сполученні параметри-значення і параметри-змінні (**`var`**-параметри).

Приклад запису заголовка процедури:

```
procedure FindMaxMin(x, y: Integer;
                     var max, min: Integer);
```

У чому ж відмінність між двома видами формальних параметрів? Якщо перед формальним параметром підпрограми відсутній який-небудь специфіка-

тор, мають справу з **передачею параметрів за значенням**. Цей метод полягає в тому, що при звертанні до підпрограми значення фактичного параметра записується у відповідний формальний параметр. Як тільки починається виконання процедури або функції, жодні зміни значення формального параметра не впливають на значення відповідного фактичного параметра. Такий спосіб діє за умовчанням, тобто у всіх випадках, коли не зазначено інший спосіб.

Використання **`var`**-параметра забезпечує **передачу параметра за адресою**. У цьому випадку в процедуру (або функцію) пересилається вже не значення фактичного параметра, а його місце розташування в пам'яті (адреса). Якщо формальний параметр має атрибут **`var`**, то будь-які зміни формального параметра будуть відбиватися в значенні фактичного параметра, оскільки вони тепер мовби займають одну і ту саму ділянку пам'яті. Насправді в цьому випадку при звертанні до формального параметра відбувається звертання до ділянки пам'яті, на яку він посилається; а оскільки цією ділянкою пам'яті є та, яку займає фактичний параметр, відбувається оперування (у тому числі й змінювання) фактичним параметром.

Важливою особливістю **`var`**-параметрів є те, що замість них як фактичні параметри можна підставляти тільки змінні (але не константи або вирази). Це, природно, потрібно враховувати при написанні програм.

Якщо виклик функції може з'являтися в будь-якому виразі, то цього не можна сказати про процедуру. Щоб звернутися до процедури, необхідно згадати її ім'я разом зі списком фактичних параметрів як **окремий оператор**.

Якщо необхідно вийти з процедури або функції, не досягши її кінця, то застосовують процедуру без параметрів `Exit`. Перед виконанням процедури `Exit` **`var`**-параметри і (при примусовому виході з функції) змінна `Result` повинні дістати значення.

Процедурами є й опрацьовувачі подій. Правда, до опрацьовувачів подій звичайно не звертаються спеціально, оскільки вони викликаються автоматично.

Схожими з **`var`**-параметрами у плані реалізації та застосування є так звані **`out`**-параметри. Якщо параметр підпрограми використовується тільки для передачі результатів роботи підпрограми в точку виклику і при цьому не використовується для передачі даних ззовні усередину підпрограми, то його можна описати з попереднім зарезервованим словом **`out`**:

```
procedure FindMaxMin(x, y: Integer;
                     out max, min: Integer);
```

При виклику підпрограми замість **`out`**-параметрів необхідно підставляти змінні. При цьому вони є тільки вихідними (результуючими).

Наголосимо: усе, що може виконати процедура, здійсненне і за допомогою функції (оскільки апарат **var**- і **out**-параметрів використовується й у функціях). Справедливим є і зворотне ствердження.

Локальні змінні (у тому числі й формальні параметри) створюються в пам'яті в момент активізації підпрограми і знищуються при виході з неї. Якщо підпрограма викликає іншу підпрограму, то локальні змінні тієї підпрограми, що викликається, розміщаються в пам'яті зразу після локальних змінних першої підпрограми, у результаті чого утворюється ланцюжок груп локальних змінних. Знищення локальних змінних виконується з хвоста цього ланцюжка. Ділянка пам'яті, у яку записуються формальні параметри і локальні змінні підпрограм, називається стеком. Стек не може займати більше 2147483647 байт і менше 1024 байт. За умовчанням розмір стека дорівнює 16384 байт. Якщо в програмі не використовуються підпрограми, можна зменшити розмір стека, а якщо вони використовуються активно, – збільшити.

Керування розміром стека здійснюється за допомогою директиви компілятору `$M`, що є директивою з параметрами. Її перший параметр задає мінімальний розмір стека, а другий – максимальний розмір (наприклад: `{$M 10000, 2000000}`). За умовчанням ця директива задана у вигляді `{$M 16384, 1048576}`.

У Delphi введено так звані константні параметри:

```
procedure proc(const param: Real);
```

Особливістю **const**-параметрів є те, що фактичні параметри, які підставляють замість них, передаються не за значенням, а за адресою (як і у випадку **var**-параметрів). Однак оголошення формального параметра зі службовим словом **const** забороняє зміни відповідного параметра усередині підпрограми, і поява операторів, що модифікують такі параметри, зумовлює діагностику помилки на етапі компіляції. Константні параметри можуть використовуватися як при описі процедур, так і при описі функцій. Параметри-масиви, якщо їхній вміст не змінюється усередині підпрограми, рекомендується оголошувати як константні параметри, оскільки в цьому випадку копії фактичних параметрів при звертанні не створюються, що сприяє економії пам'яті, займаної в стеку локальними змінними.

Відзначимо, що якщо формальний параметр має який-небудь файловий тип, то він повинен бути описаний зі службовим словом **var** (можливе вживання одного зі слів **out** або **const**).

1.2. Випереджальний опис

Ще один момент у використанні процедур і функцій – можливість випереджального опису, який полягає в тому, що використовувана підпрограма може бути описана тільки заданням свого заголовка, зразу за яким йде стандартна директива **forward**. Опис тексту такої підпрограми може бути розташований в будь-якому місці розділу описів без повторення списку формальних параметрів (хоча його можна і повторити). Випереджальний опис потрібен у випадку взаємного звернення процедур або функцій одна до одної:

```
procedure first(x, y: Real); forward;     //Випереджальний
опис

procedure second(x, y: Integer);    //Початок процедури
second
begin
          //Текст процедури second
  first(-1, 2.7);                        //Виклик процедури
first
          //Продовження тексту процедури second
end;                                     //Кінець процедури
second

procedure  first;                  //Опис тексту процедури
first
begin
          //Текст процедури first
  second(-7, 12);                        //Виклик процедури
second
          //Продовження тексту процедури first
end;                                      //Кінець процедури
first
```

Можна вважати, що методи класів, у тому числі опрацьовувачі подій, конструктори та деструктори описуються з випереджальним описом, щоправда, без зазначення службового слова **forward**.

2. ПРИКЛАДИ ПРОГРАМ

Приклад 1. Дано дійсний масив, що містить не більше 20 елементів. Домножити всі елементи масиву на його мінімальний елемент. Визначити дві підпрограми – пошуку мінімального елемента масиву і множення елементів масиву на дійсне число.

Розв'язання. Розмістимо на формі однорядковий редактор Edit1 для введення чисел (у його властивість Text запишемо число 1), багаторядковий редактор Memo1 для виведення результату та кнопку Button1 (у її властивість Caption запишемо текст Ввести). Над редактором Edit1 розмістимо мітку, записавши у її властивість Caption підказку Уведіть N. Перед описом класу TForm1 опишемо тип

```
T20Real = array [1..20] of Real;
```

а у секції **public** опису класу форми впишемо два рядки

```
N: Integer;
a: T20Real;
```

визначивши відповідно змінні для позначення розміру масиву і самого масиву.

У верхній частині секції **implementation** модуля перед кодом опрацьовувача події OnClick кнопки Button1 опишемо у зазначеному нижче порядку функцію та процедуру:

```
                                        // Визначення функції
function MinArray(const a: T20Real; Count: Integer):
Real;
var
  Min: Real;
  i: Integer;
begin
  Min := a[1];
  for i := 2 to Count do
    if a[i] < Min then Min := a[i];
  Result := Min;
end;
                                      // Визначення процедури
procedure MultArrayOnNumber(var a: T20Real;
                            Count: Integer;
                            Number: Real);
```

```
var
  i: Integer;
begin
  for i := 1 to Count do
    a[i] := a[i] * Number;
end;
```

Код оброблювача події OnClick кнопки Button1 може бути таким:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  i: Integer;
begin
  Edit1.SetFocus;
  if Tag = 0 then begin
    DecimalSeparator := '.';
    N := StrToInt(Edit1.Text);
    Label1.Caption := 'a[' + IntToStr(Tag + 1) + ']';
  end
  else begin
    a[Tag] := StrToFloat(Edit1.Text);
    Label1.Caption := 'a[' + IntToStr(Tag + 1) + ']';
    if Tag = N then begin
      Memo1.Lines.Add('Початковий масив');
      for i := 1 to N do
        Memo1.Lines.Add('a[' + IntToStr(i) + '] = '
                        + FloatToStr(a[i]));
      MultArrayOnNumber(a, n, MinArray(a, n));
      Memo1.Lines.Add('Результуючий масив');
      for i := 1 to N do
        Memo1.Lines.Add('a[' + IntToStr(i) + '] = '
                        + FloatToStr(a[i]));
      Button1.Visible := False;
      Label1.Visible := False;
      Edit1.Visible := False;
    end;
  end;
  Tag := Tag + 1;
end;
```

Приклад 2. Скласти процедуру, що видаляє з одновимірного цілочислового масиву всі числа, які входять у другий масив.

Розв'язання. Розмістимо на формі однорядковий редактор для введення чисел, багаторядковий редактор для виведення результатів і три кнопки. Накладемо кнопки одну на одну і розмістимо над редактором мітку, записавши у її властивість Caption підказку Уведіть N.

Внесемо такі зміни в значення властивостей компонентів

Мітка:

Name — lbOutput1

Caption — Розмір першого масиву

Однорядковий редактор:

Name — edInput1

Text — 1

Багаторядкове поле:

Name — mmOutput1

Lines — очистимо

Перша кнопка:

Name — btInp1

Caption — Масив 1

Visible — True

Друга кнопка:

Name — btInp2

Caption — Масив 2

Visible — False

Третя кнопка:

Name — btRun

Caption — Виконати

Visible — False

У визначення типу `TForm1` в секції **`public`** додамо опис трьох полів:

```
a, b: array of Integer;
na, nb: Integer;
```

Крім того, у верхній частині розділу **`implementation`** дамо випереджальний опис процедур:

```
procedure Del(var a: array of Integer; var na: Integer;
              b: array of Integer; nb: Integer); forward;
procedure OutputArray(a: array of Integer;
                      n: Integer; Memo: TMemo); forward;
```

Нижче дамо визначення власних процедур:

```
procedure Del(var a: array of Integer; var na: Integer;
              b: array of Integer; nb: Integer);
var
  i, j, c: Integer;
  Flag: Boolean;
begin
  c := 0;
  i := 0;
  while i < na do
  begin
    Flag := True;
    for j := 0 to nb - 1 do
      if a[i] = b[j] then
      begin
        Flag := False;
        Break;
      end;
    if Flag then
    begin
      a[c] := a[i];
      Inc(c);
    end;
    Inc(i);
  end;
  na := c;
end;

procedure OutputArray(a: array of Integer; n: Integer;
                      Memo: TMemo);
var
  i: Integer;
begin
  for i := 0 to n - 1 do
    Memo.Lines.Add(IntToStr(a[i]));
end;
```

Для розв’язання задачі можна скористатися такими опрацьовувачами події OnClick кнопок, які можна розташувати у будь-якій частині розділу **implementation** після випереджального опису процедур:

```
procedure TForm1.btInp1Click(Sender: TObject);
begin
  edInput1.SetFocus;
  if Tag = 0 then
  begin
    na := StrToInt(edInput1.Text);
    SetLength(a, na);
  end
  else
    if Tag <= na then
      a[Tag - 1] := StrToInt(edInput1.Text);
  Tag := Tag + 1;
  if Tag <= na then
    lbOutput1.Caption := 'Уведіть a[' +
                         IntToStr(Tag - 1) +']'
  else
  begin
    mmOutput1.Lines.Add('Масив A');
    OutputArray(a, na, mmOutput1);
    lbOutput1.Caption := 'Розмір другого масиву';
    btInp1.Visible := False;
    btInp2.Visible := True;
    edInput1.SetFocus;
    Tag := 0;
  end;
end;
```

3. ЗАВДАННЯ НА ЛАБОРАТОРНУ РОБОТУ

За час, відведений для виконання лабораторної роботи (2 академічні години), студент повинен:

1. Розробити алгоритм розв’язання задачі, запропонованої для програмування.
2. Здійснити проектування форми для функціонування розроблюваної програми.
3. Здійснити програмну реалізацію розробленого алгоритму.

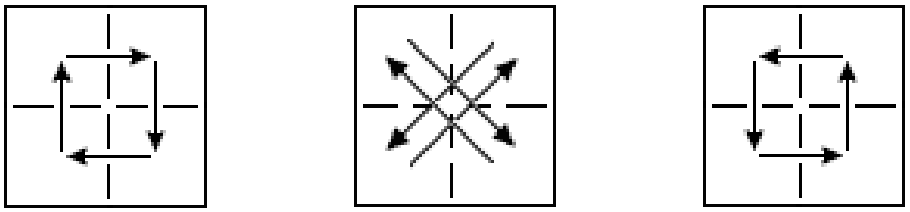
4. Здійснити відлагодження програми, виправивши синтаксичні та логічні помилки.
5. Підібрати тестові дані для перевірки програми, включаючи виняткові випадки.
6. Оформити звіт до лабораторної роботи.
7. Відповісти на контрольні запитання.
8. Здати викладачу працездатну програму з демонстрацією її роботи на декількох варіантах вихідних даних.

4. ВАРІАНТИ ЗАДАЧ

1. Дано цілі числа $p, n_0, d_0, n_1, d_1, \cdots, n_p, d_p, a, b$ $(d_0 \cdot d_1 \cdot \ldots \cdot d_p \cdot b \neq 0,\ p \geq 0)$. Обчислити $\frac{n_p}{d_p}\left(\frac{a}{b}\right)^p + \frac{n_{p-1}}{d_{p-1}}\left(\frac{a}{b}\right)^{p-1} + \ldots + \frac{n_0}{d_0}$, одержавши результат у вигляді простого дробу. Визначити процедури повного скорочення числа, заданого чисельником і знаменником, а також процедури множення та додавання простих дробів.
2. Дано одновимірний масив натуральних чисел. Видалити в цьому масиві всі прості числа. Описати підпрограми перевірки числа на те, що воно є простим, та видалення з масиву всіх простих чисел. Скористатися цими підпрограмами для рішення задачі.
3. Описати процедуру що переписує вміст масиву з n цілих чисел ($n \leq 100$) у два масиви. В один масив повинні бути записані всі парні числа, у в другий – всі непарні. Порядок слідування парних та непарних чисел повинен бути збереженим. Відлагодити підпрограму на декількох прикладах.
4. Дано натуральне число n і цілочисловий масив з n елементів. Знайти довжину найдовшої послідовності відмінних від нуля елементів масиву, що йдуть підряд. Переписати цю послідовність в інший масив. Розробити підпрограму формування вказаного масиву.
5. Дано натуральне число n і цілі числа a_1, a_2, …, a_n. Знайти відрізок масиву максимальної довжини, в якому перше число дорівнює останньому, друге – передостанньому і т. д. Визначити підпрограму одержання довжини та номера першого елемента цього відрізка (якщо їх декілька, то першого з них).
6. Описати процедуру, що виконує обмін рядків з номерами k_1 і k_2 у дійсній матриці $\mathbf{A} = (a_{ij})_{mn}$. Використовуючи цю процедуру, поміняти місцями в

заданій дійсній матриці **A** розміром $m \times n$ рядки, що містять мінімальний і максимальний елементи.

7. Описати процедуру, що виконує обмін стовпців з номерами k_1 і k_2 у дійсній матриці $\mathbf{A} = (a_{ij})_{mn}$. Використовуючи цю процедуру, поміняти місцями в заданій дійсній матриці **A** розміром $m \times n$ стовпці, що містять найбільшу і найменшу кількість додатних елементів.
8. Описати процедуру, що виконує транспонування дійсної квадратної матриці **A** порядку m без формування нової матриці. Використати цю процедуру для перетворення заданої квадратної матриці **A** порядку m у транспоновану матрицю. **Примітка.** *Транспонована матриця* формується з первинної записом рядків у вигляді стовпців (або, що те ж саме, стовпців у вигляді рядків).
9. Описати та використати у програмі процедуру для блокового переносу частин прямокутного квадратного масиву з парними розмірами відповідно до схем, наведених на рис. 4.1, *а–в*.



а *б* *в*

Рис. 4.1

10. *Сумою* двох матриць **A** і **B** розміром $m \times n$ називається така матриця **C** розміром $m \times n$, елементи якої дорівнюють сумі відповідних елементів матриць-доданків:

$$c_{ij} = a_{ij} + b_{ij},\ i = 1, 2, \ldots,\ m,\ j = 1, 2, \ldots, n.$$

Описати процедуру отримання суми двох матриць і продемонструвати її використання.

11. *Добутком* двох матриць **A** (розміром $m \times n$) і **B** (розміром $n \times p$) називається така матриця **C** (розміром $m \times p$), елементи якої дорівнюють сумі попарних добутків відповідних елементів рядка першої матриці і стовпця другої матриці:

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj},\ i = 1, 2, \ldots,\ m,\ j = 1, 2, \ldots, p.$$

Описати процедуру отримання добутку двох матриць. Процедура повинна мати параметр, призначений для передавання інформації про можливість або неможливість отримання добутку (наприклад, у цей параметр може бути записане значення –1 у разі неможливості отримання добутку і значення 0 у противному випадку).

12. Одержати масив усіх простих чисел, що не перевищують задане натуральне число n. Для розв'язання задачі описати підпрограму, яка ґрунтується на способі, що зветься «Решето Ератосфена». Суть способу в наступному. Нехай дано послідовність натуральних чисел 2, 3, …, n. Перше просте число 2. Викреслимо його і всі кратні йому числа. Перше з чисел, що залишилися, знову є простим, і можна виконати ті ж самі дії над цим числом і т. д.
13. Описати підпрограму, що перетворює дійсну матрицю $\mathbf{A}$ розміром $m \times n$, віднімаючи з рядка i_1 рядок i_2, помножений на задане дійсне число g. За допомогою цієї підпрограми, використовуючи як допоміжний той рядок, який містить максимальний за модулем елемент заданої матриці $\mathbf{A}$ розміром $m \times n$, замінити нулями в інших її рядках елементи стовпця з максимальним за модулем елементом матриці.

СПИСОК ЛІТЕРАТУРИ

1. Безменов, М. І. Основи програмування в середовищі Delphi : навч. посіб. / М. І. Безменов. – Х. : НТУ «ХПІ», 2010. – 608 с.
2. Кэнту, М. Delphi 7 : Для профессионалов / М. Кэнту – СПб. : Питер, 2004. – 1101 с.
3. Архангельский, А. Я. Программирование в Delphi 6 / А. Я. Архангельский. – М. : БИНОМ, 2002. – 1120 с.
4. Дарахвелидзе, П. Г. Программирование в Delphi 7 / П. Г. Дарахвелидзе, Е. П. Марков. – СПб. : БХВ-Петербург, 2003. – 784 с.
5. Культин, Н. Б. Основы программирования в Delphi 7 / Н. Б. Культин. – СПб. : БХВ-Петербург, 2003. – 608 с.
6. Пестриков, В. М. Delphi на примерах / В. М. Пестриков, А. Н. Маслобоев. – СПб. : БХВ-Петербург, 2005. – 496 с.
7. Ремкеев, А. А. Курс Delphi для начинающих. Полигон нестандартных задач / А. А. Ремкеев, С. В. Федотова. – М. : СОЛОН-Пресс, 2006. – 360 с.
8. Митчелл, К. Керман. Программирование и отладка в Delphi™ : учебный курс / Митчелл К. Керман. – М. : Вильямс, 2004. – 720 с.
9. Парижский, С. М. Delphi : Только практика / С. М. Парижский. – К. : МК-Пресс, 2005. – 208 с.
10. Культин, Н. Б. Основы программирования в Delphi 2007 / Н. Б. Культин. – СПб. : БХВ-Петербург, 2008. – 480 с.

Навчальне видання

Методичні вказівки
до лабораторної роботи
«Процедури у програмах мовою Delphi»
з курсу «Програмування» для студентів напряму 6.040302 – Інформатика
(спеціалізація «Соціальна інформатика»)

Укладач БЕЗМЕНОВ Микола Іванович

Відповідальний за випуск О. С. Куценко
Роботу до видання рекомендував О. В. Горелий

За авторською редакцією

План 2011 р., поз. 5/75–11

План 2011 р., поз. 5 / 75-11

Підп. до друку 23.05.2011 р. Формат 60 × 84 $^{1}/_{16}$. Папір офісний.
Riso-друк. Гарнітура Таймс. Ум. друк. арк. 0,9. Наклад 50 прим.
Зам. № 161. Ціна договірна.

Видавничий центр НТУ «ХПІ».
Свідоцтво про державну реєстрацію ДК № 3657 від 24.12.2009 р.
61002, Харків, вул. Фрунзе, 21

Друкарня НТУ «ХПІ», 61002, Харків, вул. Фрунзе, 21